PIF1006 – Mathématiques pour informaticiens II

Session Automne 2021

TRAVAIL #1

Indications principals	
Date de présentation de l'énoncé	1er octobre 2021
Date de remise du travail	29 octobre 2021
Politique sur remises tardives	-25% par jour de retard
Éléments et format de remise	Rapport et fichiers complet de la solution ou du projet remis sous forme électronique sur le portail de cours dans un seul fichier compressé (.zip)
Pondération	10%
Nb d'étudiants par équipe	1 à 5, sans exception (une équipe de 2 ou 3 est conseillée)

INTRODUCTION

Dans ce premier travail pratique, il vous sera demandé d'expérimenter avec les notions de grammaires formelles et de dérivations.

L'objectif principal du travail est de vous faire apprécier la relation qui existe entre une grammaire (incluant ses règles) et le langage qu'elle permet de générer, en plus de vous introduire à la validation syntaxique d'expressions régulières données en entrée, à la façon des grammaires formelles.

DESCRIPTION DÉTAILLÉE DU TRAVAIL À ACCOMPLIR

Vous devez créer une application console ou avec une interface graphique (à votre guise) dans laquelle un utilisateur se voit offrir différentes options qui lui permettront de créer une base de règles pour une grammaire régulière à partir d'un ensemble régulier qu'on lui fournit ou en ajoutant des règles à la base de règles manuellement.

Plus précisément, votre application doit permettre à l'utilisateur de :

- **Gérer une grammaire** : **créer/réinitialiser** une nouvelle grammaire, la **sauvegarder** sous un fichier quelconque (texte, xml, binaire, base de données, peu importe), la **charger** et lui **donner un nom**.

- **<u>Gérer les règles de la grammaire</u>** : il doit être possible de gérer les règles pour l'utilisateur en les entrant une à une dans un format du genre « A -> 1A, selon les indications du tableau ci-contre :

> **NOTE** : Le format doit emprunter une symbolique approximative de ce que nous avons vu en classe :
>
> Utilisez « 0 » et « 1 » comme terminaux, uniquement;
>
> Utilisez les lettres pour les non terminaux (A, B, C, …);
>
> Utilisez « -> », ou simplement « > » en guise de flèche représentant la dérivation;
>
> Utilisez « e » pour la chaîne vide;
>
> Évidemment vous devez vous assurer de n'accepter que les règles qui sont bien formées dans l'une ou l'autre des façons. N'oubliez pas qu'il doit y avoir au minimum une règle qui part du non-terminal S.

L'utilisateur doit pouvoir visualiser un listing de la définition de sa grammaire et des règles s'il choisit l'option.

- **Entrer une expression** dans un validateur syntaxique et **vérifier si elle fait partie de l'ensemble régulier ou non** : cette partie va demander de faire appels aux notions de structures de données et algorithmes :

  1) Vous **<u>DEVEZ</u>** utiliser les principes de « conversion » qui permettent de passer des règles de la grammaire vers des états et de transitions d'états d'un automate.

  2) **Une fois l'automate constitué** (pas besoin de le représenter nécessairement visuellement, mais au minimum une table d'instructions doit pouvoir être affichée), **l'utilisateur peut demander si certaines expressions sont reconnues ou non**. L'algorithme doit alors passer d'un état à l'autre en lisant chaque symbole un à un et voir s'il atteint un état final.

L'utilisateur se voit **afficher la réponse** « valide »/ « non valide » .

NOTE : Cela signifie que :

- Ce qui représentent les nœuds (qui sont les états) dans votre structure de données sélectionnée doivent pouvoir avoir une valeur booléenne associée qui détermine s'il s'agit d'un état final ou non

- Les liens d'un nœud vers l'autre sont unidirectionnels et peuvent être multiples.

- Si vous convertissez les règles telles quelles, l'automate pourrait bien être non déterministe et nécessiter une forme de « backtracking » pour tester plusieurs séries de transitions d'états possibles avec un même output. Facultativement, vous avez aussi la possibilité de trouver une façon de faire pour transformer votre automate non déterministe résultant en un automate déterministe équivalent, et ainsi rendre la reconnaissance facile.

  Cette page est un bon point de départ pour votre recherche :
  https://www.tutorialspoint.com/automata_theory/ndfa_to_dfa_conversion.htm

-

AUTRES EXIGENCES/INDICATIONS SUPPLÉMENTAIRES :

- Assurez-vous qu'au démarrage il soit possible de charger une base de règles de démonstrations (ou qu'elle soit chargeable à l'aide d'une option ou chargée automatiquement au démarrage). Des points seront retranchés si ce n'est pas le cas.

- Assurez-vous que vous avez des commentaires pertinents dans votre code.

- Assurez-vous de tester votre application avant de la rendre en vous assurant qu'elle peut rouler sans planter. S'il est impossible pour le correcteur de tester des options dans l'application, il n'aura pas comme mandat de débugger et recompiler votre application et les points liés au bon fonctionnement de tout ce qui en découle seront perdus.

- Assurez-vous de remettre tous les fichiers de projet/solution, les fichiers de code ET les fichiers compilés de telle sorte que le correcteur puisse ouvrir votre projet dans l'environnement de développement intégré sans devoir créer de projet à partir de vos fichiers.

- À moins d'une entente particulière, vous DEVEZ utiliser C# .Net ou Java.

RAPPORT À REMETTRE

En plus de votre solution complète, votre équipe doit remettre un rapport contenant minimalement :

- 1 page de présentation;
- 1 page détaillant le rôle joué par chacun des membres de l'équipe;
- 1 section sur les problèmes et difficultés rencontrées s'il y a lieu;
- 1 section dans laquelle vous présentez un guide utilisateur, imprime-écran à l'appui, qui non seulement détaille comment fonctionne votre application, mais aussi donne au moins un exemple de chacune des options à l'œuvre. Vous devez inclure un exemple complet d'une base de règles, de la table instructions générées par le programme conséquemment aux règles, et 2 exemples valides et non valides et le résultat donné par le programme.

GRILLE D'ÉVALUATION

Voici les critères d'évaluation et la pondération associée à chacun d'entre eux :

Sujet d'évaluation	***Pts (/10)***
Rapport complet/guide utilisateur (incl. les résultats pour les cas demandés)	1
Fonctionnalités (est-ce que ça fonctionne?): - Gestion des grammaires : 1 pt - Gestion des règles : 1.5 pts - Validateur syntaxique : 2.5 pts	5
Structures, algorithmes et forme du code (est-ce bien fait et lisible?): - Navigation entre menus / interfaces utilisateurs : 0.5 pt - Validation et formes des règles et des ensembles réguliers : 0.5 pt - Traduction règle vers état : 1 pt - Structure de données adéquates et bien exploitée pour l'automate : 1 pt - Commentaires cohérents et suffisants : 1 pt	4